VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Kalkulátor financování prodeje a pojištění automobilů formou spotřebitelského úvěru, leasingu a operativního leasingu

Calculator for Financing the Sale and Insurance of Cars in the form of Consumer Credit, Leasing and Operating Leasing

2020 Ondřej Kovács

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Ondřej Kovács**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Kalkulátor financování prodeje a pojištění automobilů formou spotřebitelského úvěru, leasingu a operativního leasingu
Calculator for Financing the Sale and Insurance of Cars in the form of Consumer Credit, Leasing and Operating Leasing

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem projektu je provést analýzu, návrh a implementaci nástroje pro výpočet spotřebitelského úvěru, leasingu a operativního leasingu pro účely vizualizace průběžných změn ceny produktu dle jeho aktuální konfigurace. Nástroj musí nabízet možnost integrace libovolné automobilky a jejího modelu osobního i nákladního vozu s modelovými sazbami a metodami výpočtu financování na základě vstupních parametrů definovaných uživatelem.

1. Trendem dnešní doby je při prodeji nových a ojetých vozů kalkulovat cenu jako celek nebo pravidelné měsíční splátky. Ty jsou ovlivněny řadou faktorů jako jsou akontace, doba splácení, doplňkové služby (povinné ručení, havarijní pojištění) atd.
2. Student provede analýzu současných trendů v oblasti vizualizace alternativ financování při prodeji automobilů, analyzuje a implementuje komponentu, která na základě předdefinovaných sazebníků bude počítat přibližnou výši úvěru a splátek při prodeji konkrétního vozu v prodejních informačních systémech.
3. Student sestaví nástroj pro výpočet leasingových splátek na základě definovaných kritérií.
4. Na základě spolupráce s vybranou finanční institucí provede analýzu dostupných API pro získávání sazeb, eventuálně celých výpočtů splátek a pojištění.
5. Výsledkem práce bude analýza, návrh a implementace systému pro kalkulaci financování automobilu, včetně návrhu vhodné struktury pro jejich ukládání a přepočet na základě změněných parametrů financování v reálném čase.
6. Výsledkem bude teoretická příručka, která pomůže při volbě vhodné technologie pro daný projekt s ohledem na výkon, jednoduchost nasazení a finanční náročnost.
7. Teoretické poznatky budou ověřeny na modelovém příkladu s reálnými daty. Výsledné řešení bude porovnáno s existujícími kalkulátory a dostupnými API.

Seznam doporučené odborné literatury:

[1] GORMLEY, Clinton a Zachary TONG. Elasticsearch: the definitive guide. ISBN 1449358543.
[2] SHKLAR, Leon. a Rich. ROSEN. Web application architecture: principles, protocols and practices. 2nd ed. Hoboken, NJ: Wiley, c2009. ISBN 047051860x.
[3] SHIVAKUMAR, Shailesh Kumar. Architecting high performing, scalable and available enterprise web applications. ISBN 9780128022580.
[4] WEERAWARANA, Sanjiva. Web services platform architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and more. Upper Saddle River, NJ: Prentice Hall PTR,

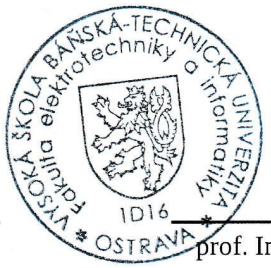
c2005. ISBN 0131488740.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Radoslav Fasuga, Ph.D.**

Datum zadání: 01.09.2019

Datum odevzdání: 30.04.2020



doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna 2020

.....

Souhlasím se zveřejněním této bakalářská práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TUO Ostrava.

V Ostravě 30. dubna 2020 .....

Rád bych poděkoval vedoucímu bakalářské práce Ing. Radoslavu Fasugovi, Ph.D. za odbornou pomoc a konzultaci při vytváření této práce. Také bych chtěl poděkovat společnosti UniCredit Leasing CZ, a. s. za vysvětlení problematiky financování automobilů a dále společnosti Generali Česká pojišťovna a.s. za osvětlení problematiky pojišťování vozů.

Abstrakt

Cílem bakalářské práce je implementace kalkulátoru financování prodeje a pojištění automobilů. Byla provedena analýza již hotových řešení. Práce obsahuje návrh a implementaci kalkulátoru, který zpracovává veškeré nutné parametry pro výpočet a následně je zobrazuje do uživatelského rozhraní. Součástí tohoto řešení je i volba vhodných technologii pro finální implementaci. Výsledkem mé bakalářské práce je webová aplikace, která spolupracuje na pozadí s API rozhraním a data zobrazuje do jednoduchého webového rozhraní.

Klíčová slova: PHP, Nette, kalkulátor, financování, HTML5, jQuery, MySQL, relační databáze, MVP, AJAX

Abstract

The goal of this bachelor thesis is the implementation of car sale and insurance financing calculator. Analysis of completed solutions was carried out. The work contains the design and implementation of a calculator, which processes all the necessary parameters for the calculation and then displays them in the user interface. This solution also includes the selection of suitable technologies for the final implementation. The result of my thesis is a web application that cooperates with API interface in the background and displays data into a simple web interface.

Keywords: PHP, Nette, calculator, financing, HTML5, jQuery, MySQL, relation database, MVP, AJAX

Obsah

Seznam použitých zkratek a symbolů

MVP	–	Model-View-Presenter
PHP	–	PHP: Hypertext Preprocessor
CSS	–	Cascading Style Sheets
HTML	–	Hyper Text Markup Language
DPH	–	Daň z přidané hodnoty
URL	–	Uniform Resource Locator
UX	–	User experience
Nette	–	PHP framework
AjAX	–	Asynchronous JavaScript and XML

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Při pořizování vozidla, lze využít různé možnosti financování. Snad každá značka aut poskytuje na svých webových stránkách online konfigurátor vozidla, kdy v posledním kroku bývá možnost vypočítání financování vozidla. Lze upravovat různé parametry a tím ovlivňovat měsíční splátku. Ovšem ne každý si chce pořídit nový vůz a internet nám neposkytuje dostatečné množství kalkulátorů, kde bychom si mohli spočítat a zobrazit si podrobnosti financování. Máme možnost využít například srovnávače úvěrů či pojištění automobilu, ale tím přijdeme o možnost si různě upravovat parametry tak, aby nám financování vyhovovalo co nejvíce. Z toho důvodu jsem se rozhodl vytvořit univerzální konfigurátor, který je lze použít pro všechny značky nových i ojetých vozů. V posledním kroku, po vzoru profesionálních konfigurátorů značek automobilového průmyslu, se nachází přehledný a detailní kalkulátor, na kterém lze měnit parametry financování vozidla, ale i pojištění.

Úkolem bakalářské práce bylo navrhnout a vytvořit funkční kalkulátor financování a pojištění vozidel. Nejprve bylo nutné najít a porovnat existující řešení, dále jsem zanalyzoval problematiku celého kalkulátoru. Díky těmto krokům, jsem dokázal vybrat ty nejvhodnější funkce a ty následně použil pro vývoj mého kalkulátoru.

Implementaci jsem rozdělil do tří plně propojených částí. První část je klientská – jedná se o responzivní konfigurátor obsahující kalkulačku financování a pojištění s důkladem na UX. Druhá je interní API rozhraní, které zpracovává klientské požadavky pro výpočet financování, ty ukládá do cache pro zrychlení odbavení požadavků. Data dále posílá Presenteru, kde se zpracují a zobrazí uživateli. Poslední část je administrace nabízející kompletní správu konfigurátoru, včetně sazeb pro kalkulačku.

2 Analýza problematiky

2.1 Financování automobilu

Při pořizování automobilu si lze zvolit různé možnosti financování. Málo kdo má u sebe naspořenou celou částku pro pořízení automobilu, proto existují různé alternativy, jak si vysněné auto zakoupit.

2.1.1 Úvěr od banky

Jako první možnost se nabízí půjčka, kterou lze zprostředkovat různými způsoby. Peníze si lze půjčit například od banky. Pokud se jedná o úvěr na automobil, povětšinou dojde k převodu financí přímo na účet prodejce, nebo bazaru, a vypůjčitel pouze splácí sumu, kterou mu banka poskytla. Druhou možností je si půjčit tzv. na cokoliv, kdy se peníze převedou na účet vypůjčiteli a je možné je využít nejenom ke koupi auta.

2.1.2 Úvěr od prodejců

Snad každý z autorizovaných prodejců (např. Škoda, Volkswagen), ale i autobazary (např. AAA auto), mají své možnosti financování. Kdy peněžní prostředky poskytne vypůjčiteli vybraná leasingová společnost, se kterou má daný prodejce smlouvu. Například formou úvěru, kdy klient zaplatí akontaci, splátku předem, (není vždy potřeba, záleží od prodejce) z celkové částky vozidla a zbytek peněz poskytne úvěrová společnost, tato částka se splácí v měsíčních splátkách. Určité procento prodejců poskytují standardní půjčku, kdy se splácí měsíčně konstantní částka, která se určila při úvodní kalkulaci úvěru.

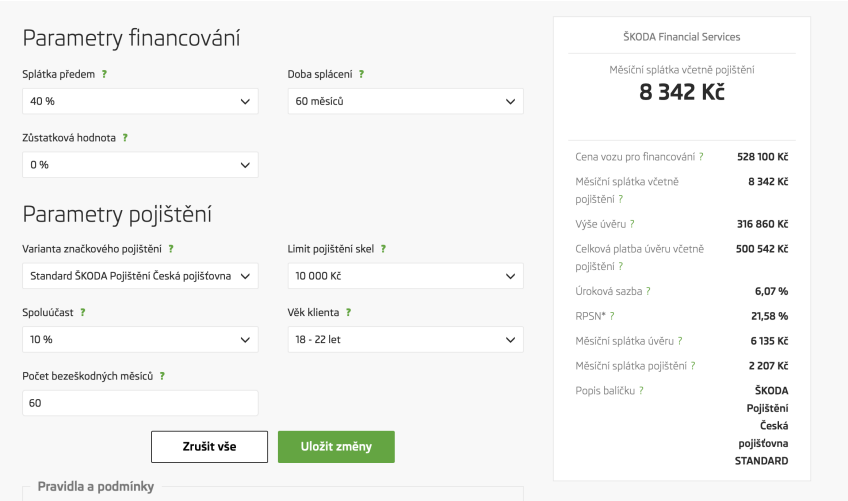
2.1.3 Financování poslední nerovnoměrnou splátkou

Na druhou stranu, poměrně hodně značek poskytuje speciální úvěr, tzv. poslední nerovnoměrné splátky, kdy se po dobu například 5 let splácí úvěr v menších částkách a poslední finální splátka je vysoká. Jedná se o sumu, kterou je potřeba splatit do finální ceny vozu. Výhodou tohoto speciálního úvěru je možnost si vybrat, co při poslední splátce udělat. Lze ji buďto v plné výši doplatit, nebo existuje možnost nechat si připravit nové financování na další roky a zaplatit ji po částech (ve splátkách). Poslední možností je výměna automobilu. Společnost odkoupí stávající vozidlo a suma získaná touto transakcí pokryje výše zmíněnou poslední nerovnoměrnou splátku. Zbylé prostředky se použijí při splácení nového (vyměněného) vozu.

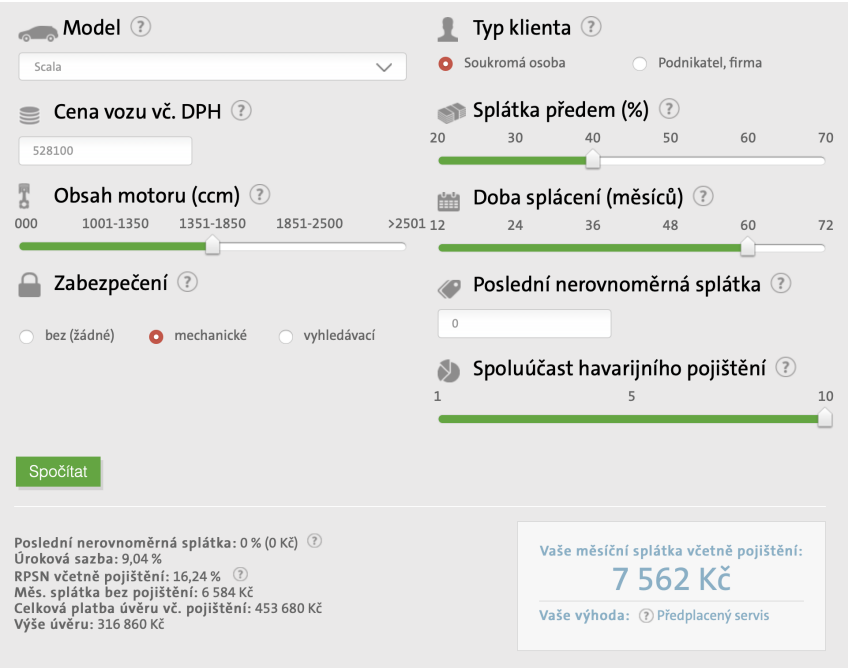
2.1.4 Příklad financování úvěrem od prodejců

Při konkrétním příkladu jsme si vzali novou Škodu Scala s celkovou hodnotou vozu 528 100 Kč. Vyžádali jsme si možnost financování poskytovaného přímo společností Škoda v jejich konfigurátoru[1] (viz. Obrázek 1) a stejný vůz v stejné hodnotě u Volkswagen Financial Services[2]

(Obrázek 2). Výsledek byl překvapivý, na každé webové stránce vycházela jiná měsíční splátka, tedy i jiný úrok a RPSN, popřípadě cena pojištění. Ale ta je relativní, jelikož u konfigurátoru Škody, probíhá podstatně větší segmentace klienta. Z toho důvodu je pojištění vozu i měsíční splátka vozidla přesnější.



Obrázek 1: Výpočet financování v konfigurátoru Škoda



Obrázek 2: Výpočet financování na stránkách VW Financial Services

2.1.5 Operativní leasing

Další možností financování automobilu je operativní leasing. Tuto možnost platby především využívají firmy a malí podnikatelé, pro které se jedná o konstantní měsíční náklady, jež si můžou navést do účetnictví. Současně se jedná o největší lákadlo, nového vozu každý rok. Postupem času jej začali využívat i některé fyzické osoby, přesto většina uživatelů této služby jsou právě právnické osoby a OSVČ. Tuto možnost financování poskytují, jak různí autorizovaní prodejci automobilů, tak i společnosti, které právě otevírají způsob této platby pro fyzické osoby.

2.1.6 Srovnání

Oproti běžnému úvěru v bance má operativní leasing jisté výhody. Největší z nich je právě amortizace vozidla a dále také povinné ručení a havarijní pojištění obsažené v měsíční splátce. Při úvěru z banky je sjednání pojištění nutné osobně, což činí další finanční náklady při pořizování vozidla. Další výhodou často bývají do ceny započítané pravidelné servisy, nebo zimní/letní pneumatiky v ceně měsíční splátky. Na druhou stranu při financování auta úvěrem, se stáváme majitelem vozidla ihned, při operativním leasingu nikdy. Jsme pouze provozovatelem po dobu, kdy se operativní leasing platí. U tématu vlastnictví, je praktické představit si i klasický leasing, v tomto případě jsme provozovatelem do doby, než se vozidlo splatí celé a pak jej můžeme za symbolickou částku odkoupit a majitelem vozidla se stáváme my.

Obrovskou nevýhodou při operativním leasingu ovšem je, omezení nájezdu na kilometry. Kontrola automobilu na konci pronájmu, kdy je povětšinou potřeba doplatit poškozená místa na vozidle, například poškrábané sedačky.

Výhodou úvěru může být možnost si pořídit automobil z bazaru, tím pádem nemusí být nutnost si pořizovat nové auto. Tímto zamezíme přílišné amortizaci vozu, kdy vůz ztrácí s každým kilometrem na hodnotě, a to nejvíce právě během prvních let. Autobazary navíc často nabízejí bonus formou vykoupení starého vozu, nebo slevy na autě v případě, že si vůz klient zakoupí na splátky.

Pokud zakoupíme vůz přes autorizovaného prodejce, lze také povětšinou získat slevu, ale i další dodatečné bonusy, například to, že povinné ručení a havarijní pojištění už je součástí splátky. Pokud vůz celý zaplatíme, stáváme se jeho výhradními majiteli.

2.1.7 Pojištění vozidla

Dle zákona je povinnost mít auto pojištěné alespoň povinným ručením. Tuto službu poskytují snad všechny různé pojišťovny, ale každá z nich má jiné nabídky služeb, které si lze přikoupit. To samozřejmě potom ovlivňuje konečnou cenu pojištění. Povinné ručení lze platit třemi způsoby, a to jednou ročně, nebo půlročně, či čtvrtletně. Některé pojišťovny mají cenový rozdíl při způsobu platby čtvrtletně oproti třeba ročnímu. Pokud si vynásobíme čtvrtletní platby, kolik by nás vyšly za rok, vyjde nám podstatně větší částka, než kdybychom si zvolili platbu roční. Takto to má

vyřešené například Kooperativa pojišťovna, a.s., Vienna Insurance Group. Zcela jiný přístup má například Česká pojišťovna a.s.

V případě pořízení ojetého vozu se doporučuje vyřídit si havarijní pojistku. Ta kryje klienta v případě, kdy způsobí nehodu a poškodí se i jeho automobil. Pojistník je povinen zaplatit určité procento, které je dohodnuté ve smlouvě, z opravy jeho vozidla. Zbytek peněz, do celkové opravy, doplatí pojišťovna. Při velké a drahé nehodě se to zcela jistě hodí, pokud by se jednalo o nehodu menší, pojišťovna je vždy krytá minimální částkou, kterou klient musí zaplatit, například 10 tisíc korun při 10 % spoluúčasti. Cena havarijního pojištění se odvíjí od hodnoty vozu.

2.2 Financování a pojištění bydlení

2.2.1 Financování

Spotřebitelské úvěry na bydlení lze rozdělit do dvou skupin. První skupinu charakterizuje potřeba dát něco do zástavy, nějakou nemovitost. Jedná se o klasickou hypotéku nebo o typ tzv. americké hypotéky (úvěr na cokoliv). Do druhé skupiny patří úvěr ze stavebního spoření, kde se do určité výše nevyžaduje ručitel ani zastavování majetku.

2.2.2 Klasická hypotéka

Klasická hypotéka je nejběžnější volbou pro lidi, kteří potřebují získat peníze na nové bydlení, popřípadě rekonstrukci bydlení. Má dost nízký úrok – běžně do 3 %, závisí to na bance, fixaci a délce splacení. Díky nízkému úroku je pro klienta ideální volbou, ale k jejímu získání je nutné mít čím ručit, popř. mít ručitele a lze ji využít pouze ve souvislostí s bydlením.

2.2.3 Americká hypotéka

Americká hypotéka se od té klasické liší možností vzít si ji na cokoliv, nemusí se jednat pouze o bydlení. Na úkor toho má horší úrok – průměrně okolo 4-5 %, to závisí na bance, fixaci a délce splacení. V tomto případě je také nutnost mít ručitele, nebo dát nemovitost do zástavy.

2.2.4 Úvěr ze stavebního spoření

Úvěr ze stavebního spoření má dvě odnože, první klasický úvěr ze stavebního spoření a druhý překlenovací úvěr (tzv. meziúvěr). U první možnosti je nutnost mít naspořeno alespoň 40-50 % z cílové částky, splňovat minimum hodnoticího čísla a spořit alespoň 2 roky. Tyto podmínky neplatí u druhého, překlenovacího úvěru. Úroková sazba je horší v případě překlenovacího úvěru, jelikož jsou podmínky volnější. V obou případech je úvěr účelový a jedná se vždy pouze o bydlení (rekonstrukce nebo koupě).

2.2.5 Výpočet měsíční splátky

Orientační výpočet měsíční splátky se počítá následujícím vzorcem, viz. 1

$$s = \frac{D\left(1+\frac{p}{100}\right)^n \frac{p}{100}}{\left(1+\frac{p}{100}\right)^n - 1} \tag{1}$$

kde,
D – hodnota úvěru poskytnutého bankou
p – úroková míra
n – délka úvěru

Úrokovou mírů (p) nám ovlivňuje fixace, výška úvěru i celková délka. Dále samozřejmě různé provize, které si uzmou banky.[3]

2.2.6 Splácení úvěru

Splácení hypotečního úvěru závisí na způsobu jeho čerpání. Úvěr lze čerpat jednorázově nebo postupně. Pokud je hypoteční úvěr čerpán postupně, splácí se nejprve měsíčně pouze úrok z vyčerpané částky, a to až do doby, než je vyčerpán celý úvěr. Po ukončení čerpání začne dlužník hypoteční úvěr splácet většinou anuitními splátkami, které jsou po celou dobu platnosti úrokové sazby stejné a obsahují jistinu a úrok.

Některé banky umožňují i jiné splácení než splácení anuitními splátkami. Jedná se o degresivní nebo progresivní splácení tzn., že velikost splátek je v čase buď klesající (degresivní) nebo rostoucí (progresivní).

V termínu, kdy dochází ke změně úrokové sazby (klient si volí v letech dobu platnosti úrokové sazby), může dlužník splatit předčasně část, nebo celou výši úvěru bez sankce. V případě mimořádné splátky ze strany klienta mimo tyto termíny si většina bank stanovuje vysoké sankce za předčasné splácení. V případě neúčelové hypotéky, která spadá pod režim zákona o spotřebitelském úvěru (č. 321/2001 Sb.), lze kdykoli učinit mimořádnou splátku části nebo celého úvěru.[4]

2.2.7 Porovnání

Nejvýhodnější financování, v souvislosti s bydlením, má klasická hypotéka, poskytuje nejvýhodnější úrok, ale je nutnost mít čím ručit. Následuje americká hypotéka, zde je úrok sice horší, ale o to větší výhoda si půjčit na cokoliv, také je nutnost ručit. Pro ty, co nemají čím ručit zbývá úvěr ze stavebního spoření, který má platební podmínky ze všech 3 případů nejhorší.

2.2.8 Pojištění nemovitostí

Při pojištění domácnosti lze zohlednit dvě skupiny rizik. První je riziko krádeže a druhé živelné zkázy. Pojistit lze movité věci v domácnosti s ní související a nadále příslušenství a stavební

součásti přiléhající k vnitřním prostorám bytu. Pojistné plnění a podmínky určuji pojišťovny. Existuje také možnost dokoupit i další doplňkové připojištění, jako tomu je i u automobilů (obecně na věci, na které se nevztahuje běžné pojištění).

2.3 Srovnávače a kalkulačky

Na internetu existuje mnoho stránek, které srovnávají různé půjčky, hypotéky, ale i pojištění. Tyto stránky mohou být nezávislé nebo mohou mít dohodnutou spolupráci s konkrétní firmou, kterou upřednostňují více než ty ostatní. Tak či onak, nabízejí kompletní přehled všech různých typů půjček bank, dle zadaných preferencích – počet měsíců, úrok apod. Popřípadě pokud nabízí některá z bank akci, tak se tento fakt zohlední a doporučí jako výhodnější řešení. Mezi tyto služby patří například stránky `www.usetreno.cz`, `www.penize.cz` a `www.mesec.cz`.

Dále někdy některé stránky, včetně těchto uvedených, nabízí i články kde se věnuji této problematice – půjčky a hypotéky a dávají svým klientům cenné rady. Některé zase nabízejí možnost zastoupení a finančního poradenství. Proto je vhodné zvážit o kterou službu se konkrétně zajímáme.

Některé srovnávače pojištění (viz. Tabulka 1) mohou dokonce rovnou, díky spolupráci s pojišťovnami, nabídnout výhodné pojištění. Mezi takovéto srovnávače patří například nejznámější webová stránka `Klik.cz`[5], nebo méně známější `www.povinne-ruceni.com`[6], `ePojisteni.cz`[7], nebo `www.srovnator.cz/povinne-ruceni/`[8] . Tyto služby se těší rostoucímu zájmu, jelikož si na internetu stačí navolit parametry vozidla a poté se s Vámi dealer (zprostředkovatel) spojí. Telefonicky s Vámi prodiskutuje detaily ohledně vozidla a segmentaci klienta, nabídne nejlepší pojišťovnu z cenového hlediska, a navrhne se smlouva. Zákazníkovi už pouze zbývá včas zaplatit, a to do pár dnů od založení pojistky, jinak je pojistka neplatná.

Tabulka 1: Srovnání webů podle počtu pojišťoven

Web	**Počet pojišťoven**
Klik.cz	12
ePojisteni.cz	9
povinne-ruceni.com	15
Srovnator	15

Ihned je k dispozici k vytisknutí zelená karta, pojištění tedy nabývá platnosti s ukončením telefonátu. Smlouva přijde pár dnů od zaplacení běžnou, nebo elektronickou poštou. Taková služba má určitě pro některé jistou výhodu, a to vyřízení všeho z pohodlí domova. Není ovšem pravidlem, aby taková forma pojištění patřila k nejvýhodnějším na trhu – tyto společnosti musí z něčeho vydělávat, účtují si proto různé poplatky za zpracování, vyřizování nebo provize z pojistky.

Při zjišťování výhod jednotlivých srovnávačů jsem odhalil, že Povinne-ruceni a Srovnator jsou ve výsledku stejné stránky. Mají stejnou infolinku a výsledky na zadaný požadavek jsou

identické. Dokonce vzhled stránek (struktura) je identický, pouze se jemně liší rozložení a barevné schéma. Výsledky cen se liší hlavně u společnosti Klik.cz, ta udává nejvyšší možnou cenu, ostatní weby lákají na nižší rozmezí ceny. Všechny weby vyžadují telefonní číslo za účelem konzultace, popřípadě poskytnutí slev, a právě tehdy se může lišit výsledná částka.

3 Návrh implementace

3.1 Kalkulace financování

Jako konzultační firmu pro moji práci, která má na starost financování, jsem si zvolil společnost UniCredit Leasing CZ, a. s., která na trhu poskytuje u většiny automobilek v ČR různé financování (finanční leasing, operativní leasing nebo úvěr). Jmenovitě se jedná o značky Hyundai, Dacia, BMW, Renault, Honda, Mazda a další.

Po kontaktování nemalého množství lidí jsem dostal konečně kontakt na business development managera, který má na starost řešení této problematiky, a který mi pomohl osvětlit, jak to u nich alespoň zhruba funguje. Například, jak funguje akce s 0% úrokem na úvěr automobilu, v tomto případě musí automobilka doplatit úvěrové společnosti provize.

Dále mi pomohli vytvořit vzorec, viz. 2, který budu používat pro orientační výpočet měsíční splátky:

$$\textit{měsíční splátka} = (f - a)\,\frac{q^n\,(q-1)}{q^n - 1} \tag{2}$$

kde,
f – znamená financovaná hodnota vozu,
a – akontace, kterou zákazník může poskytnout,
n – doba splacení v měsících
q – úroková míra za časovou jednotku $\left(1 + \frac{\frac{s+d+p}{100}}{12}\right)$,
s – úroková sazba v procentech,
d – dotace v procentech,
p – provize v procentech,

Další informaci, kterou jsem se při konzultaci dozvěděl je, že existuje přímá úměra prémiovosti značky a celkové ceny vozu s výši úroku (včetně dotací a provizí). Z video hovoru jsem zjistil, že společnost UniCredit Leasing CZ, a. s. poskytuje svým partnerům interní API rozhraní, které má ve svém vnitru implementované sazebníky. Jelikož se jedná o firemní tajemství nedostal jsem možnost toto rozhraní používat, tím pádem přesnost úroků bude čistě orientační. Dále bude bez provizní a dotační složky, ačkoliv tyto částky by se daly také nasimulovat.

Kdybychom chtěli co nejpřesnější procento úroku, museli bychom zohledňovat hodnotu peněz a celkový finanční trh. ČNB totiž průběžně aktualizuje úroková procenta, tím pádem bych bychom museli kalkulačku aktualizovat, alespoň měsíčně. Proto jsem se rozhodl jejich sazebník, v mé práci nahradit čtyřrozměrným polem (první dimenze značí model vozidla, druhá dimenze cenu vozidla, třetí dimenze splátku předem, a nakonec čtvrtá dimenze dobu splacení). Tímto řešením alespoň trochu napodobím funkčnost jejich interního sazebníku. Pro moji práci bude toto řešení dostačující.

Zjistil jsem, že společnosti zpravidla mají svého smluvního pojišťovatele, který jim je schopen nabídnout lepší ceny pojištění, než může pojišťovna nabídnout fyzické osobě. Další výhodou úvěru přes autorizované prodejce může být třeba i to, že málo která společnost segmentuje zákazníky do takového stupně jako to dělají pojišťovny a díky tomu třeba můžeme získat podstatně lepší pojistku.

Pro praktickou implementaci mého řešení, jsem potřeboval zjistit alespoň orientační úroky. Finanční kalkulačka společnosti UniCredit Leasing CZ, a. s. je s pevným procentuální úrokem, proto jsem nucen zkusit jiný kalkulátor. Vyhledal jsem si další finanční společnost na trhu, a to Volkswagen Financial Services (SkoFIN s.r.o.)[9], jejich kalkulátor poskytoval přesnější úroky.

Nejdříve mě napadlo využít neveřejné API rozhraní stránky vwfs.cz, která poskytuje financování pro značky jako např. Škoda, Volkswagen, Audi, Seat, Porsche a další. Bohužel při testování této API, jsem zjistil, že pro nějaké značky, je API chybná, nevrací žádné výsledky, popřípadě v horším případě vrací chyby. Další mínus této implementace je závislost na neveřejné API. Kdyby náhodou ji nějak změnili, moje aplikace by mohla přestat fungovat.

Nakonec jsem se rozhodl, že se budu zaměřovat pouze na značku Škoda, ta na stránkách vwfs.cz fungovala v pořádku. Vygeneroval jsem si jednoduchým skriptem, který odkazuje na URL API s jednotlivými parametry, sazebník, který jsem si uložil do databáze. Ke všemu díky použití tohoto off-line sazebníku bude má aplikace nezávislá na jejich API a současně použitelná i bez internetu.

3.2 Kalkulace pojištění

Kontaktoval jsem jednu z největších pojišťoven v ČR, a to konkrétně Generali Česká pojišťovna a.s., ohledně spolupráce. Jelikož jsem již vypočítávání měsíční splátky ceny vozidla vyřešil, bylo potřeba zajistit kalkulaci pojištění – povinného ručení a havarijní pojistky do měsíční splátky automobilu.

Po konzultaci s pracovníkem Generali Česká pojišťovna a.s. mi byla přiblížena problematika pojišťování a rozdíl mezi tím, když pojišťuje leasingová společnost oproti pojišťovně, nebo třeba serverům jako Klik.cz, a další. Leasingové společnosti ve svém sazebníku neřeší, nebo to dělají zcela minimálně, segmentaci. Z toho plyne, pokud chceme pořídit nový vůz určitě se více oplatí řešit koupi vozu pomocí úvěru (leasingu) u autorizovaného prodejce. Můžeme totiž získat levnější pojistku, pokud nemáme vysoké bonusy. Dále mi osvětlil výpočet havarijní pojistky, kdy každá značka má jiné procento úroku, podle prémiovosti vozidla. Existuje konstanta pro spoluúčast, kterou si určí také každá značka.

K mému projektu využiji starší sazebník pojištění povinného ručení, který jsem získal od konzultanta z Generali Česká pojišťovna a.s., a pro výpočet havarijního pojištění zvolím pevnou úrokovou sazbu pro značku Škoda 4,5 % a pro procentuální spoluúčast volím konstanty zapsány v Tabulce 2.

Tabulka 2: Konstanty spoluúčasti

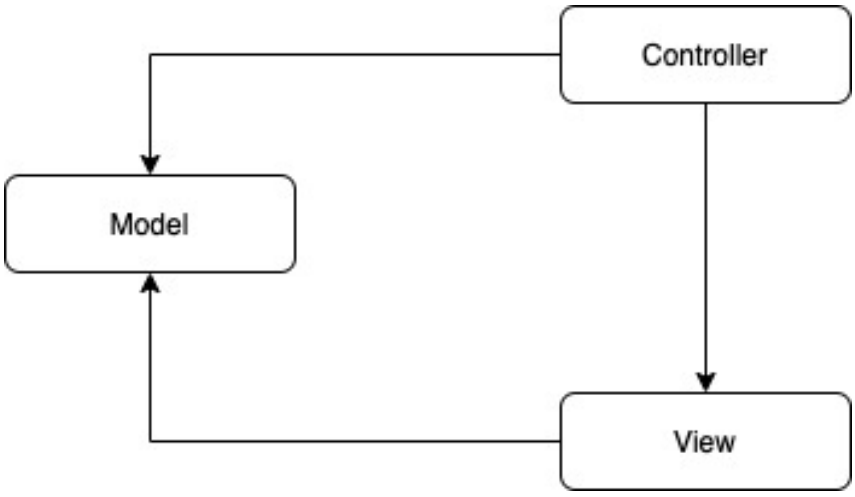
Spoluúčast	**Konstanta**
1% spoluúčast	1.4
5% spoluúčast	1.2
10% spoluúčast	1

3.3 Popis řešení

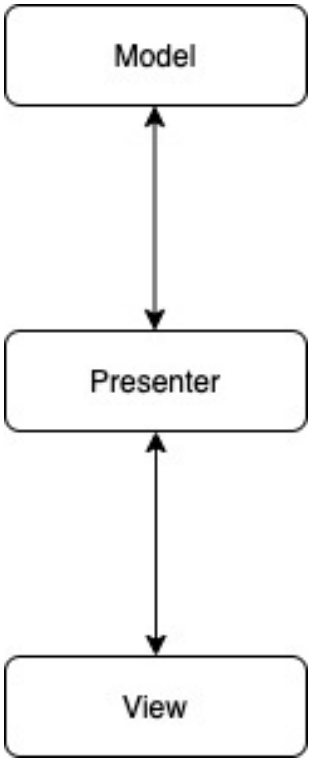
Kalkulátor bude fungovat na webových stránkách s důrazem na responzivitu. Díky tomu, uživatelé budou mít přístup ke kalkulátoru jak z počítače, tak i z telefonu. Kalkulátor bude naprogramován v jazyce PHP ve verzi 7.4.2, který se používá pro vývoj internetových aplikacích.[10] Jako velkou výhodu oproti ASP.NET je multiplatformnost, kdy nám vznikají nižší náklady na server, jelikož můžeme využít servery s Linuxem. Abychom si ulehčili práci s programováním využijeme pro implementaci kalkulátoru Nette Framework.

3.3.1 Nette Framework

Jedná se o český Framework, který byl vyvinut Davidem Grudlem. Dbá především o zabezpečení aplikace. Ošetřuje aplikaci například před XSS (Cross-site scripting), Cross-Site Request Forgery. Dále dokáže ochránit před SQL injection.[**sqlInjection**] Dále obsahuje, různé užitečné knihovny, pro práci s formuláři, s routerováním a například databázi a další jiné knihovny. Aktuální verze Nette Frameworku je 3.0, ta podporuje až PHP verzi 7.4 a vyžaduje alespoň PHP 7.1. Nette Framework je založen na architektuře MVP (Model – View – Presenter) – viz. Obrázek 4 , která vychází z MVC (Model – View – Controller) – viz. Obrázek 3. Hlavní rozdíl je v tom, že presenter nahrazuje controller. Presenter se stará o zajištění dat do view vrstvy a současně zpracovává signály z view vrstvy.[11]



Obrázek 3: Architektura MVC



Obrázek 4: Architektura MVP

3.3.2 Databáze

Pro uchování sazebníku jsem zvolil MySQL databázi. Nette Framework má podporu pro tento databázový systém, tím pádem vývoj aplikace půjde efektivněji. Jelikož MySQL poskytuje i bezplatnou licenci, tak nebude problém najít webhosting pro aplikaci. Pro případnou správu databáze využiju grafické rozhraní Adminer, který mě osobně přijde jednodušší a přehlednější než phpMyAdmin.

3.3.3 Composer

Composer jednoduchými příkazy dokáže spravovat PHP projekt a jeho všechny závislosti a knihovny, jako je právě například i Nette Framework. Jako správu projektu si můžeme představit například stažení knihovny, která není v projektu obsažena, ale je nutná pro jej běh nebo například zajišťování aktuálních verzí knihoven a závislostí. V Composer.json se určují veškeré závislosti, tento soubor lze editovat manuálně nebo pomocí příkazů. Composer.lock určuje všechny verze závislostí a ty jsou stahovány do složky vendor.[12]

3.3.4 Vizualizace aplikace

Kalkulačka musí být responzivní, jelikož mnoho uživatelů na weby a webové aplikace přistupuje z tabletů a mobilních telefonů. Pro zvýšení efektivity vývoje, jsem využil Framework, který pomáhá psát aplikaci responzivně. Jedná se o Bootstrap, vyvinut původně pro Twitter. Dále kalkulačka bude zpracovávat uživatelské požadavky pomocí AJAXu. Díky asynchronním odbavení požadavků a následně překreslení Nette snippetů, dosáhneme uživatelsky přívětivým operacím.

3.3.5 Bootstrap

Bootstrap je Framework pro psaní responzivních webů a webových aplikacích. Obsahuje mnoho vytvořených komponent například pro tlačítka, formuláře, slidery a mnoho dalších, kdy stačí pouze aplikovat na HTML prvek určitou třídu a bootstrap již zbytek pomocí svých CSS na JavaScriptů zařídí. Využívá grid systém, rozkládá šířku stránky na 12 sloupců a pomocí příslušných tříd určovat, jak se budou zobrazovat na webu. Díky speciálním třídám, lze šířku ovlivnit pouze na určité breakpointy stránky (jako například, pokud má stránka maximálně 576 px). Tím docílíme různých responzivních rozloženích na různých zařízeních jen pomocí tříd HTML elementů.[13]

3.3.6 jQuery

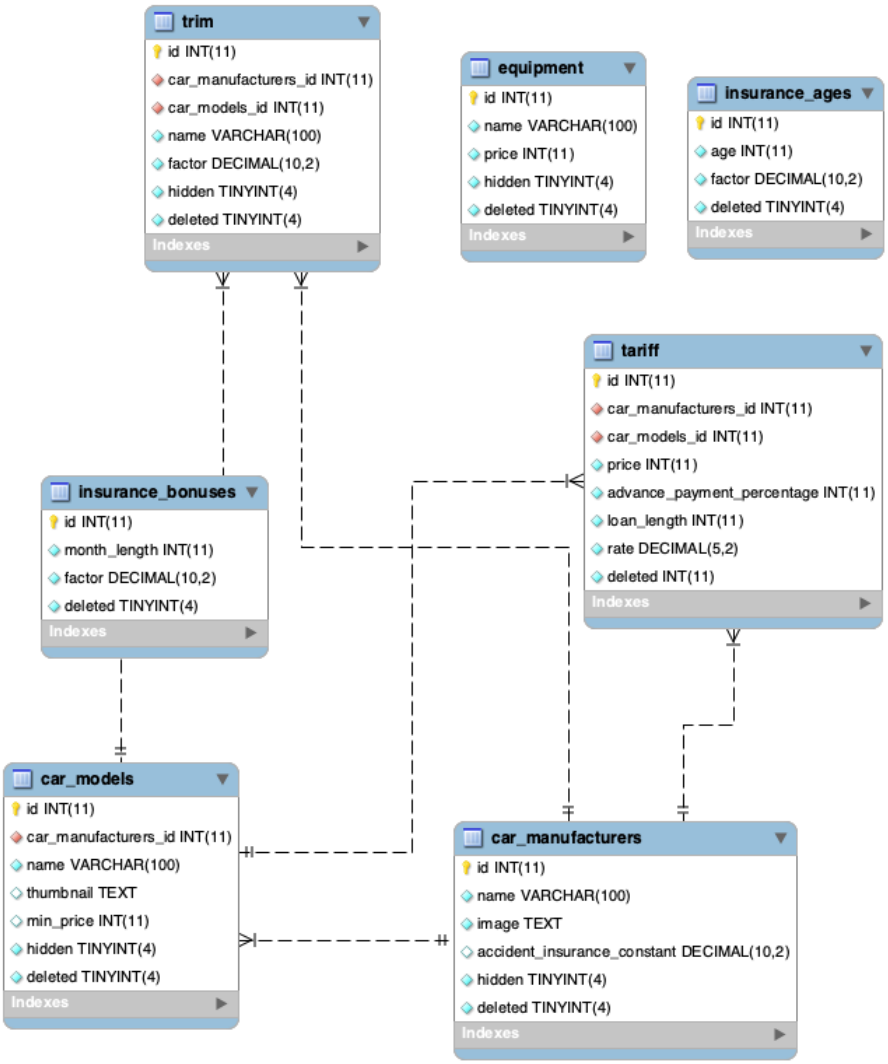
Jedná se o JavaScriptovou knihovnu, která klade důraz na interakci mezi JavaScriptem a HTML. jQuery se snaží oddělovat chování aplikace od HTML. Dále přináší plno zjednodušení pro práci např. s AJAXem, manipulaci se styly, eventy nebo výběr DOM elementů. Aktuální verze je 3.4.1. Bootstrap taky potřebuje pro svoji funkčnost právě jQuery, proto jsem se rozhodl, ho využít dále pro svoji kalkulačku.[14]

4 Datová a funkční analýza

Tato kapitola rozebírá uchovávání dat v MySQL databázi. Dále obsahuje lineární zápis tabulek a datový slovník. V datovém slovníku jsou detailněji popsány sloupce tabulek z hlediska pojmenování sloupce, datového typu (*varchar*, *int*, *decimal* apod.), velikosti datového typu, jestli se jedná o nějaký klíč, dále také jestli může být *NULL*, indexy sloupců a stručný popis funkce daného sloupce. V druhé části se popisují jednotlivé funkce aplikace. Tento popis obsahuje nastínění toho jak funguje, vstupní a výstupní data a k čemu slouží.

4.1 Datová analýza

4.1.1 Doménový model



Obrázek 5: Doménový model aplikace

4.1.2 Lineární zápis

Vysvětlivky: **Primární klíč**, cizí klíč
car_manufacturers(**id**, name, image, accident_insurance_constant, hidden, deleted)
car_models(**id**, car_manufacturers_id, name, thumbnail, min_price, hidden, deleted)
tariff(**id**, car_manufacturers_id, car_models_id, price, advance_payment_percentage, loan_length, rate, deleted)
equipment(**id**, name, price, hidden, deleted)
trim(**id**, car_manufacturers_id, car_models_id, name, factor, hidden, deleted)

4.1.3 Datové slovníky

Vysvětlivka:

1. Klíče:
   (a) PK – Primární klič
   (b) FK – Cizí klič

Tabulka 3: Datový slovník tabulky car_manufacturers

Název	Typ	Velikost	Klíč	Null	Index	Popis
id	int	11	PK	Ne	Ano	Identifikace
name	varchar	100	-	Ne	Ano	Název značky
image	text	-	-	Ne	Ne	Logo značky
accident_insurance_constant	decimal	(2,10)	-	Ano	Ne	Konstanta pro výpočet havarijního pojištění
hidden	int	4	-	Ne	Ano	Skrytý záznam
deleted	int	4	-	Ne	Ano	Smazaný záznam

Tabulka 4: Datový slovník tabulky car_models

Název	Typ	Velikost	Klíč	Null	Index	Popis
id	int	11	PK	Ne	Ano	Identifikace
car_manufacturers_id	int	11	FK	Ne	Ano	Identifikace značky auta
name	varchar	100	-	Ne	Ano	Název modelu auta
thumbnail	text	-	-	Ano	Ne	Obrázek modelu
min_price	int	11	-	Ano	Ne	Minimální cena modelu
hidden	int	4	-	Ne	Ano	Skrytý záznam
deleted	int	4	-	Ne	Ano	Smazaný záznam

Tabulka 5: Datový slovník tabulky tariff

Název	Typ	Velikost	Klíč	Null	Index	Popis
id	int	11	PK	Ne	Ano	Identifikace
car_manufacturers_id	int	11	FK	Ne	Ano	Identifikace značky auta
car_models_id	int	11	FK	Ne	Ano	Identifikace modelu aut
price	int	11	-	Ne	Ano	Minimální částka vozidla
advance_payment_percentage	int	11	-	Ne	Ano	Procento platby předem
loan_length	int	11	-	Ne	Ano	Délka úvěru
rate	decimal	(2,5)	-	Ne	Ano	Úroková sazba
deleted	int	4	-	Ne	Ano	Smazaný záznam

Tabulka 6: Datový slovník tabulky equipment

Název	Typ	Velikost	Klíč	Null	Index	Popis
id	int	11	PK	Ne	Ano	Identifikace
name	varchar	100	-	Ne	Ano	Název doplňkové výbavy
price	int	11	-	Ne	Ne	Cena výbavy
hidden	int	4	-	Ne	Ano	Skrytý záznam
deleted	int	4	-	Ne	Ano	Smazaný záznam

Tabulka 7: Datový slovník tabulky trim

Název	Typ	Velikost	Klíč	Null	Index	Popis
id	int	11	PK	Ne	Ano	Identifikace
car_manufacturers_id	int	11	FK	Ne	Ano	Identifikace značky auta
car_models_id	int	11	FK	Ne	Ano	Identifikace modelu aut
name	varchar	100	-	Ne	Ano	Název stupně výbavy
factor	int	11	-	Ne	Ne	Konstanta zvyšující cenu vozidla
hidden	int	4	-	Ne	Ano	Skrytý záznam
deleted	int	4	-	Ne	Ano	Smazaný záznam

4.2 Funkční analýza

4.2.1 Administrace

Funkce přihlášení administrátora
Tato funkce slouží k přihlášení uživatele do administračního rozhraní. Funkce je implementována jako formulář, kde je potřeba zadat uživatelské a heslo. Pokud uživatel zadá existující a platné údaje je přesměrován do administrace.

Use case 1
Název: přihlášení administrátora
Cíl: přihlásit uživatele do administračního rozhraní
Účastník: Administrátor

1. Aplikace vyzve uživatele k zadání uživatelského jména a hesla
2. Uživatel zadá údaje
3. Aplikace odešle formulář
4. Aplikace zkontroluje jestli jsou údaje správné
   (a) Aplikace zjistí, že jsou údaje nesprávné nebo uživatel nemá přístup do administračního rozhraní
   (b) Aplikace vypíše chybovou systémovou hlášku
   (c) Aplikace vyzve uživatele k opětovnému přihlášení
5. Aplikace přihlásí uživatele
6. Aplikace přesměruje uživatele do administračního rozhraní
7. Aplikace vypíše hlášku, že byl uživatel úspěšně přihlášen

Funkce přidání nové značky
Do aplikace lze přidat, v administračním rozhraní, nové značky. Ty se pak zobrazují v konfigurátoru, pokud nejsou skryté nebo smazané. Funkce je implementovaná jako formulář, kde je potřeba pouze zadat název značky a konstantu pro havarijní pojištění. Logo značky lze po té nahrát při editaci v detailu značky.

Use case 2
Název: přidání nové značky
Cíl: přidání do konfigurátoru novou značku
Účastník: Administrátor

1. Uživatel přejde v administračním rozhraní do karty Značky a modely
2. Uživatel klikne na zelené tlačítko Nová značka
3. Aplikaci zobrazí ve vyskakovacím okně formulář pro přidání značky
4. Uživatel vyplní formulář
5. Uživatel odešle formulář
6. Aplikace zkontroluje jestli konstanta pro havarijní pojištění je validní float
   (a) Aplikace zjistí, že údaj nesplňuje validační pravidlo
   (b) Aplikace vypíše upozornění
   (c) Aplikace vyzve uživatele k opravení údaje
7. Aplikace přidá značku do databáze
8. Aplikace vypíše hlášku, že značka byla úspěšně přidána

Funkce přidání nové úrokové sazby
Do kalkulátoru je možnost nové úrokové sazby. Ty pak pomáhají přesnějšímu výběru sazby při počítání měsíční splátky. Funkce je implementovaná pomocí in-line formuláře , jako nový řádek datagridu. Po úspěšném vytvoření kalkulátor úspěšně pracuje se zadanou sazbou.

Use case 3
Název: přidání nové úrokové sazby
Cíl: přidání nové úrokové sazby pro kalkulátor
Účastník: Administrátor

1. Uživatel přejde v administračním rozhraní do karty Správa úroků
2. Uživatel klikne na tlačítko "+"
3. Aplikaci vykreslí nový řádek v datagridu, kde bude moct uživatel vyplnit formulář
4. Uživatel vyplní formulář
5. Uživatel odešle formulář stisknutím modrého tlačítka save
6. Aplikace zkontroluje jestli jsou zadány povinné a validní data
   (a) Aplikace zjistí, že údaj nesplňuje validační pravidlo nebo nejsou vyplněné povinná pole formuláře
   (b) Aplikace vypíše upozornění

(c) Aplikace vyzve uživatele k opravení údajů

7. Aplikace přidá novou úrokovou sazbu do databáze
8. Aplikace vypíše hlášku, že úroková sazba byla úspěšně přidána

Funkce editace doplňkové výbavy
Pro potřebu změny ceny , nebo názvu doplňkové údaje, lze tak učinit v administračním rozhraní aplikace. Funkce je implementována jako in-line editace pole datagridu, které se automaticky zaktualizuje při překliku z pole.

Use case 4
Název: editace doplňkové výbavy
Cíl: změna údajů doplňkové výbavy
Účastník: Administrátor

1. Uživatel přejde v administračním rozhraní do karty Správa doplňkové výbavy
2. Uživatel klikne na řádek a dané pole, které chce změnit
3. Aplikace překreslí pole na editovatelné
4. Uživatel změní hodnotu překresleného pole
5. Uživatel odešle formulář stisknutím kdekoliv mimo editované
6. Aplikace zkontroluje jestli je zadaná hodnota validní
   (a) Aplikace zjistí, že údaj nesplňuje validační pravidlo,např. nejedná se o stejný datový typ
   (b) Aplikace vypíše upozornění
   (c) Aplikace vyzve uživatele k opravení hodnoty
7. Aplikace aktualizuje řádek v databázi o novou hodnotu
8. Aplikace vypíše hlášku, že doplňková výbava byla úspěšně změněna

Funkce skrytí stupně výbavy
Pokud je potřeba skrýt určitý stupeň výbavy pro vybraný model značky vozidla, můžeme tak učinit v administračním rozhraní. Implementace této funkce je pomocí tlačítka, které aktualizuje záznam v databázi a tak se v kalkulátoru daný stupeň výbavy skryje.

Use case 5
Název: skrytí stupně výbavy
Cíl: skrytí stupně výbavy v kalkulátoru před klientem
Účastník: Administrátor

1. Uživatel přejde v administračním rozhraní do karty Správa stupně výbavy
2. Uživatel vybere stupeň výbavy v datagridu, který chce skrýt
3. Uživatel klikne na žluté tlačítko na konci řádku
4. Uživatel změní hodnotu překresleného pole
5. Aplikace zaktualizuje záznam v databázi, tak že mu nastaví sloupec hidden na hodnotu 1
6. Aplikace aktualizuje řádek v datagridu a vykreslí nové zelené tlačítko, pro opětovné zobrazení
7. Aplikace vypíše hlášku, že stupeň výbavy byl úspěšně skryt

4.2.2 Konfigurátor s kalkulátorem

Funkce zvolení značky
První krok konfigurátoru je volba značky, pro kterou chceme spočítat financování. Volba probíhá klikem na kartu značky, v které je název a pokud je vyplněno logo, tak i logo značky. Po kliku se odešle AJAX požadavek, kde odpověď požadavku jsou modely dané značky, pokud nějaké existují.

Use case 6
Název: zvolení značky
Cíl: zvolení značky vozidla pro výpočet financování vozidla
Účastník: Uživatel

1. Uživatel klikne na kartu značky
2. Aplikace odešle AJAX požadavek
3. Aplikace uloží do session id značky, kterou uživatel vybral
4. Aplikace získá z databáze všechny dostupné modely pro vybranou značku
5. Aplikace vypíše modely, které byly vráceny z požadavku aplikace
6. Aplikace odemkne další krok

7. Aplikace přejde do dalšího kroku konfigurátoru, kde již jsou vypsané modely

Funkce zvolení modelu

Druhý krok konfigurátoru je výběr modelu, pro kterou chceme spočítat financování. Pokud neexistuje žádný dostupný model, který je vložen v aplikaci, musí uživatel se vrátit zpět na první krok a vybrat jinou značku. V případě, že jsou dostupné modely, lze zvolit kliknutím na kartu modelu a tím se přejde do dalšího kroku konfigurátoru. Po kliknutí se odešle AJAX požadavek, který vrátí dostupné stupně výbavy zvolené kombinace modelu a značky.

Use case 7

Název: zvolení modelu
Cíl: zvolení modelu vozidla pro výpočet financován vozidla
Účastník: Uživatel

1. Uživatel klikne na kartu modelu
2. Aplikace odešle AJAX požadavek
3. Aplikace uloží do session id modelu, kterou uživatel vybral
4. Aplikace získá z databáze všechny dostupné stupně výbavy pro zvolenou kombinaci značky a modelu
5. Aplikace vypíše stupně výbavy, které byly vráceny z požadavku aplikace
6. Aplikace upraví ceny stupně výbavy pomocí koeficientu a minimální částkou modelu, tím vznikne minimální cena vozidla s danou výbavou
7. Aplikace odemkne další krok
8. Aplikace přejde do dalšího kroku konfigurátoru, kde již jsou vypsané dostupné stupně výbavy

Funkce zvolení stupně výbavy

Třetí krok konfigurátoru je zvolení stupně výbavy, další povinný krok pro výpočet financování. Pokud neexistuje žádný dostupný model, který je vložen v aplikaci, musí uživatel se vrátit zpět na druhý krok a vybrat jiný mode. Na klik na stupeň výbavy se odešle AJAX požadavek, kde se uloží vybrané data a vypíšou případné doplňkové výbavy a barvy vozidla.

Use case 8

Název: zvolení stupně výbavy
Cíl: zvolení stupně výbavy pro výpočet financování vozidla
Účastník: Uživatel

1. Uživatel klikne na kartu stupně výbavy
2. Aplikace odešle AJAX požadavek
3. Aplikace uloží do session id stupně výbavy a také zvolenou cenu, která se zobrazuje u stupně výbavy
4. Aplikace získá z databáze všechny dostupné doplňkové výbavy
5. Aplikace vypíše doplňkovou výbavu, které byly vráceny z požadavku aplikace
6. Aplikace odemkne další krok
7. Aplikace přejde do dalšího kroku konfigurátoru, kde jsou vypsány doplňkové výbavy a volba barvy vozidla

Funkce úprava stylu vozidla
Předposlední krok konfigurátoru je úprava stylu vozidla, to znamená dobrovolný výběr doplňkové výbavy a hlavně povinný výběr barvy. Doplňkovou výbavu lze vybrat kliknutím na kartu, nebo také kliknutím na checkbox v rohu karty. Barvu zle vybrat klikem na barevné políčko, tím se karta označí. Pro vypočtení financování se klikne na kalkulovat. Po té se zobrazí v posledním kroku kalkulátor s financováním.

Use case 9
Název: úprava stylu vozidla
Cíl: zvolení doplňkové výbavy a zvolení barvy vozidla
Účastník: Uživatel

1. Uživatel může kliknou na kartu doplňkové výbavy
   (a) Aplikace zkontroluje jestli je již karta označená nebo ne, pokud není označená tak se bude cena zvyšovat o cenu doplňkové výbavy, pokud je ta se bude odečítat
   (b) Aplikace provede danou operaci v závislosti na označení karty
   (c) Aplikace aktualizuje cenu uloženou v session
   (d) Aplikace překreslí celkovou cenu vozidla, aby se aktualizovala po provedenou aplikaci
2. Uživatel vybere barvu vozidla
3. Aplikace označí barevnou kartu za vybranou
4. Uživatel klikne na tlačítko kalkulovat
5. Aplikace zkontroluje, jestli uživatel vybral povinný parametr barvu

(a) Aplikace zjistí, že barva není vybraná

(b) Aplikace vypíše upozornění

(c) Aplikace vyzve uživatele k vybrání barvy

6. Aplikace naplní selectboxy hodnotami z databáze a vybere jako výchozí ty, které vybral uživatel

7. Aplikace provede výpočet financování a vypíše výsledky

8. Aplikace odemkne další krok

9. Aplikace přejde do posledního kroku konfigurátoru, kde se nachází kalkulátor a informace o financování a vybrané konfiguraci

Funkce kalkulace financování
Poslední krok konfigurátoru je samostatný kalkulátor, který obsahuje i sumarizaci o konfiguraci vozu a informace o financování. Parametry jako značka, model a cena vozidla jsou neměnné, lze změnit pouze pomocí jednotlivých kroků konfigurátoru. V kalkulátoru lze měnit parametry pojištění a nebo délkou či akontací úvěru. Každou změnu v kalkulátoru je nutno potvrdit stiskem na tlačítko vypočítat. Tehdy se překreslí všechny detaily o financování.

Use case 10
Název: kalkulace financování
Cíl: výpočet financování se zvolenou konfigurací
Účastník: Uživatel

1. Uživatel vybere parametry financování

2. Uživatel klikne na tlačítko vypočítat

3. Aplikace zkontroluje jestli jsou všechny povinné parametry zvoleny

4. Aplikace zkontroluje jestli již není uložen výsledek financování v cache

   (a) Aplikace odešle dotaz na api s příslušnými parametry vozidla

   (b) Aplikace uloží výsledek api do cache

      i. Aplikace zjistí, že nastala při zpracování chyba, nebo chyběly potřebné parametry pro kalkulaci

      ii. Aplikace odstraní chybný výsledek z cache, pokud je již uložen

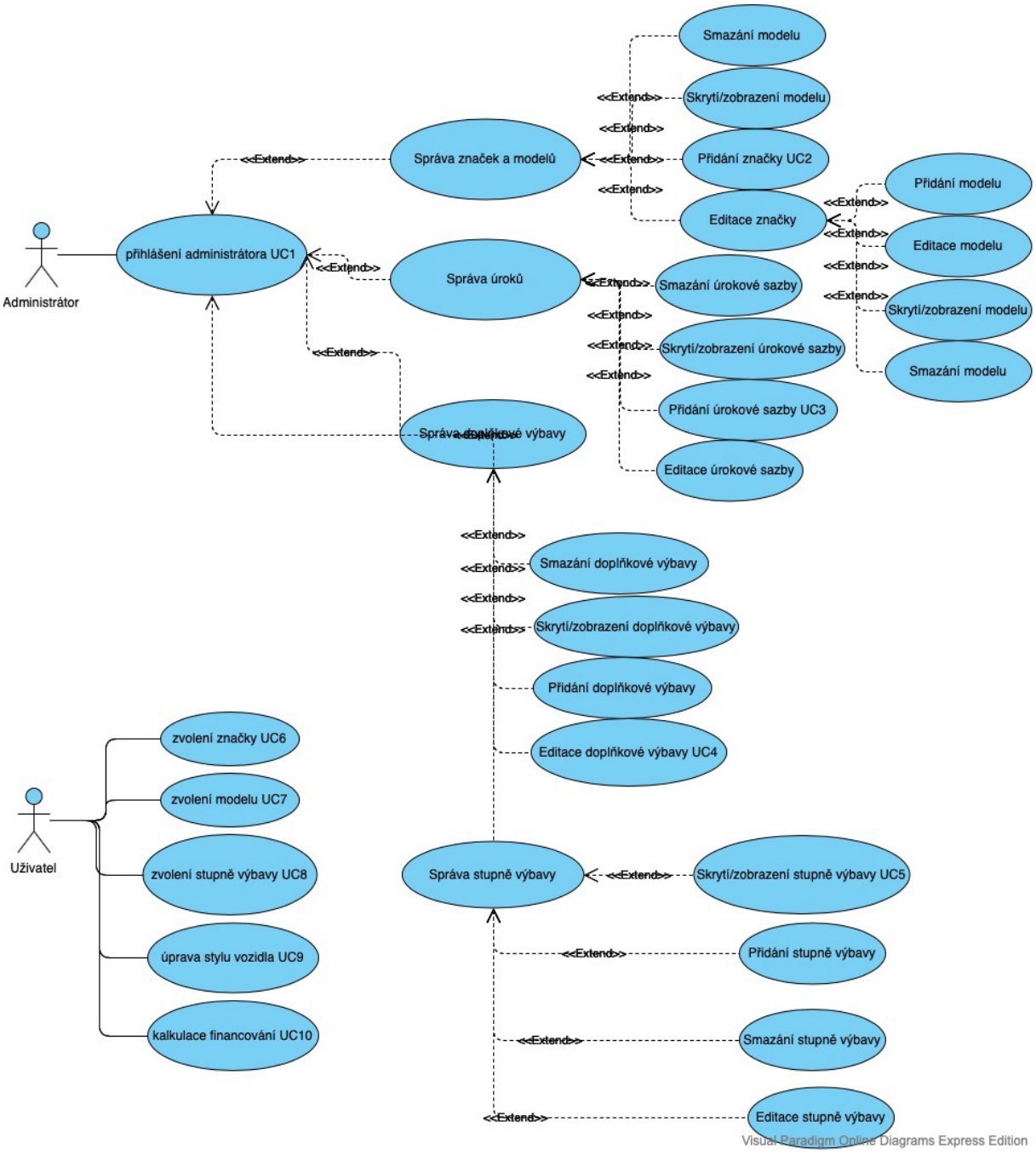
      iii. Aplikace vyzve uživatele k opětovnému zpracování

   (c) Aplikace vrátí výsledky financování

5. Aplikace zaktualizuje konfiguraci vozidla v šabloně

6. Aplikace zaktualizuje informace o financování vozidla v šabloně

4.2.3 Use case diagram



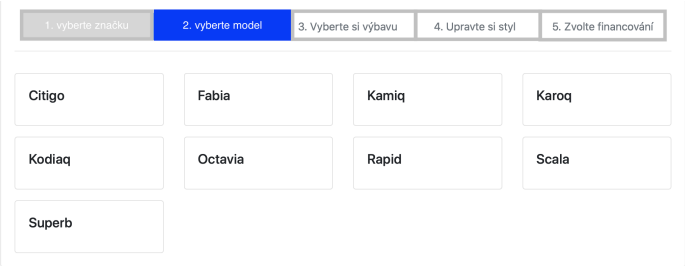
Obrázek 6: Use case diagram

5 Návrh uživatelského rozhraní

Při implementaci kalkulátoru a administračního rozhraní jsem vycházel z drátěného modelu (tzv. wireframe). Návrh se rozděluje na dvě sekce, kalkulátor a administrační rozhraní.

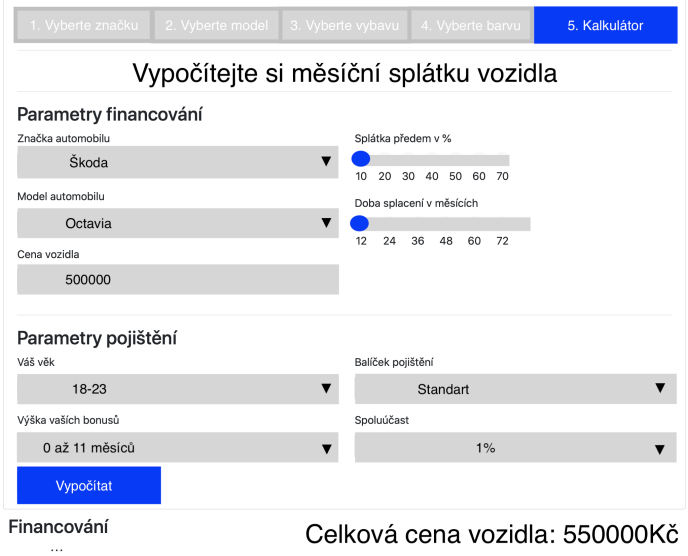
5.1 Kalkulátor

Hlavní bod aplikace je konfigurátor s kalkulátorem (Obrázek 7 a 8), ten uvidí hned po najetí na webovou stránku. Struktura konfigurátoru se nijak nemění, jedná se o 5 kroků, kdy každý je umístěn v okně karty. Nejvíce odlišný krok je finální – kalkulátor (8). Všechny kroky karty jsou responzivní pro většinu nejpoužívanějších typů rozlišení zařízení.



Obrázek 7: Koncept vzhledu konfigurátoru

Tento koncept znázorňuje jeden z kroků konfigurátoru, podobný vzhled budou mít i všechny ostatní kroky kromě posledního. Jednotlivé prvky (na Obrázku 7 zrovna modely) jsou vypsány jako klikací karty, které provedou další potřebnou činnost a přesunou do dalšího kroku. Na vrchu konfigurátoru lze nalézt předchozí, stávající a budoucí kroky barevně odlišeny.



Obrázek 8: Koncept vzhledu kalkulátoru v konfigurátoru, jako poslední krok

Na tomto obrázku 8 vidíme odlišný, poslední krok konfigurátoru a tím je kalkulátor. Kalkulátor se skládá z select boxů, textových polí, dvou posuvníku rozsahu (tzv. range sliders) a sumarizace financování. Vždy se lze vrátit k předchozím krokům a tím změnit výsledek financování.

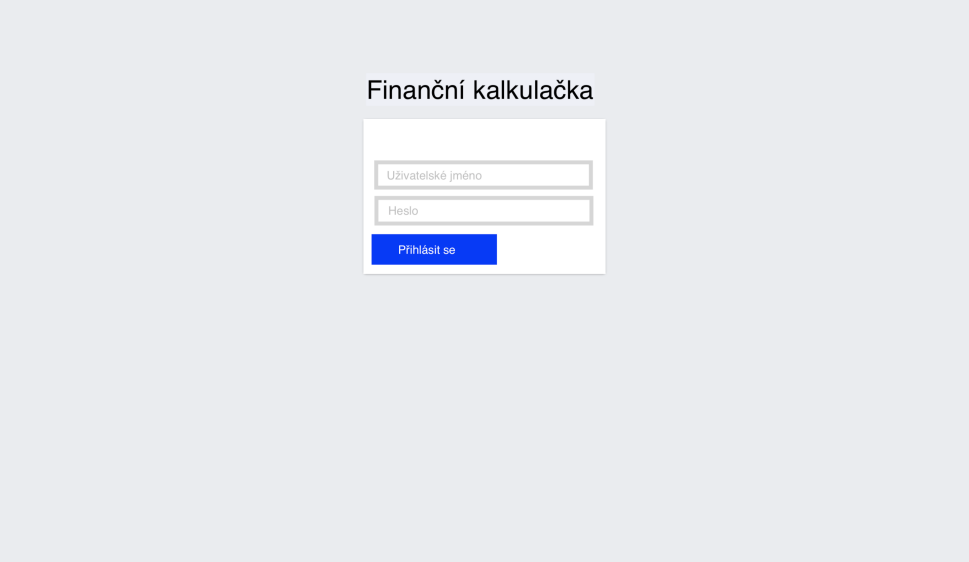
5.2 Administrační rozhraní

Do administračního rozhraní se běžný uživatel nedostane, je potřeba znát URL adresu, kde se nachází a samozřejmě také přihlašovací údaje, které zadáme v prvním kroku administrace – viz. Obrázek 9. Po přihlášení přejdeme do administrace kalkulátoru, kde můžeme spravovat různé sekce.

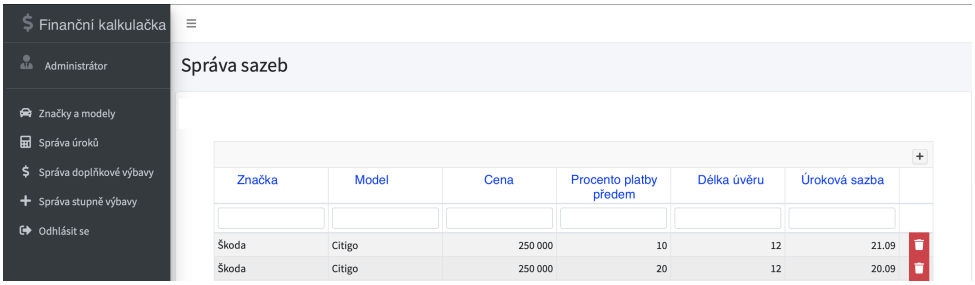
Koncept přihlášení (Obrázek 9) je velice jednoduchý. Jedná se o prostý formulář s přihlašovacím jménem a heslem. Plně responzivní a jednoduchý design.

Na dalším konceptu, obrázek 10 můžeme vidět celý layout administrace včetně menu. Menu na responzivní verzi se schová do tzv. hamburger menu, kdy po kliknutí na znak hamburgeru (3 vodorovné čáry pod sebou) vyjede zleva menu. Dále najdeme koncept datagridu pro výpis položek. V administraci budu používat 2 různé datagridy, ale o tom víc v kapitole 6 – Implementace.

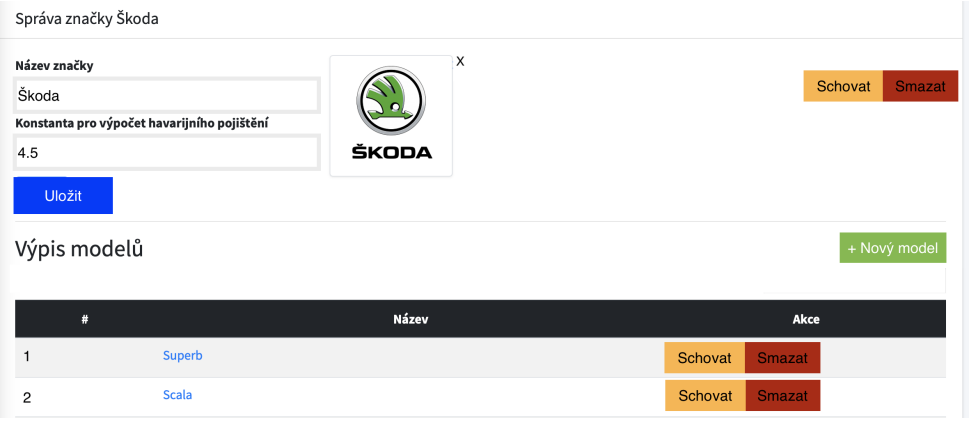
Další část, která se bude využívat v administračním rozhraní, hlavně pro značky a modely je jejich detail s editací – viz 11. Na tomto konceptu se především jedná o zaměření na detail značky. Zde totiž kromě editace dané značky se budou zobrazovat i dostupné modely s proklikem na jejich detail, popřípadě smazáním nebo skrytím z konfigurátoru. Editace polí je řešena pomocí klasického formuláře. Pokud je již nějaká fotka nahraná, lze ji pomocí tlačítka vedle "x"smazat. Po té se objeví tlačítko pro nahrání nové fotky.



Obrázek 9: První krok administrace – přihlášení



Obrázek 10: Koncept layoutu administrace včetně výpisu pomocí datagridu



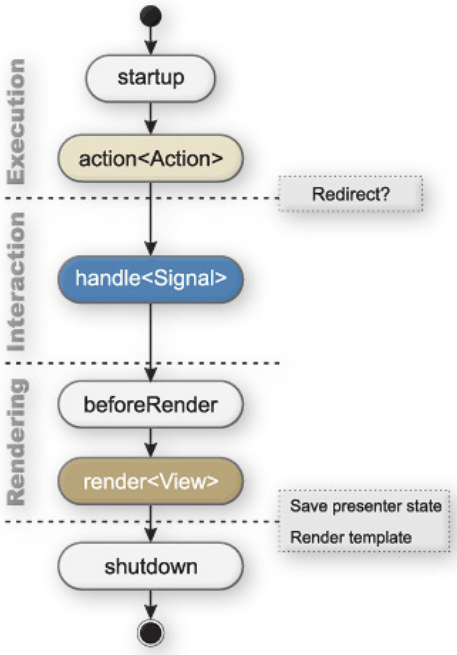
Obrázek 11: Koncept editace značky – bez ukázky layoutu

6 Implementace aplikace

Jelikož používám při programování aplikace Nette Framework, který vychází ze architektury MVP, je potřeba dodržovat strukturu i její principy. Na začátek jsem si vygeneroval tzv. skeleton Nette aplikace. To mi zajistilo správné rozložení souborů do složek. Dále jsem se rozhodl při vývoji použit verzovací systém git. Založil jsem si privátní repozitář na GitHubu, kde jsem po té vždy pushoval změny v aplikaci. Pro ulehčení práce, jsem využil pro administrační rozhraní již nakódovanou šablonu. To mi ušetřilo mnoho času, než by se taková responzivní šablona zhotovila. Použil jsem šablonu AdminLTE 3 od Colorlib, ti poskytují mnoho zdarma přístupných Boostrap šablon.

6.1 Aplikační vrstva

Jak jsem již zmínil Nette využívá MVP architekturu programování. Aplikační vrstvu zajišťují presentery, v který se řeší veškerá logika. Určitě doporučuji, při práci s Nette, znát životní cyklus presenteru (Obrázek 12). Pokud bychom ho neznali, mohlo by nastávat zbytečně k chybám, jako například přepisování dat v různých signálech. V naší aplikaci hlavně budeme pracovat ze základními sigánly **action<nazev>()**, **render<nazev>()** a **handle<nazev>()**. Při vývoji jsem si rozdělil aplikaci do dvou modulů – AdminModule a FrontModule. Díky tomu, by mohl vývoj probíhat agilněji, kdyby pracovalo více programátorů, tak by mohl každý řešit svůj modul. Další výhodou určitě je rozdělní zdrojových kódu.



Obrázek 12: Životní cyklus presenteru

Při vývoji administrace, jsem si vytvořil supertřídu pro presenter – **BasePresenter**, ta dědí z potřebné třídy **Presenter** od Nette, díky tomu, aplikace bude vědět, že se jedná o presenter nikoliv o normální třídu. Moje třída **BasePresenter** implementuje potřebné funkce, na všech stránkách – jako například odhlášení. Díky principu dědění z **BasePresenteru** všechny ostatní presentery budou implementovat všechny potřebné funkce, bez toho bych je musel zvlášť implementovat na každém presenteru. Při vývoji konfigurátoru (FrontModule) nebylo potřeba zřizovat takovou supetřídu, jelikož většina funkcí se implementovala v tom daném presenteru a nebylo potřeba dědit metody. Navíc celý konfigurátor (bez api) je napsaný v jednom presenteru – jedná se o jednu stránku tudíž nebylo potřeba řešit více presenterů.[15]

6.2 Formuláře

Pro práci s formulářem je potřeba ji nejdříve vytvořit pomocí továrny pro komponenty – tvar **createComponent<Name>()**, kde *name* je název vytvářené komponenty. Komponenta je následně připojena k presenteru. Jelikož se jedná o vytvoření komponenty obecné, ne jen formuláře, je potřeba trošku změnit syntaxi. Pro vytvoření formuláře je se používá tvar **createComponent<Name>Form()**, díky PHP 7.1 můžeme napsat, že metoda nám vrátí třídu *Form* – viz ukázka kódu 1

```
public function createComponentLoginForm(): Form
{
        $form = new Form();
        $form->addText('username', '')->setRequired("Vyplňte uživatelské jméno"
            );
        $form->addPassword('password')->setRequired("Vyplňte prosím heslo");
        $form->addSubmit('login', 'Přihlásit se');

        $form->onSuccess[] = [$this, "loginFormSuccess"];
        return $form;
}
```

Výpis 1: Vytvoření jednoduchého, přihlašovacího formuláře v Nette

Musíme také implementovat, logiku formuláře po úspěšném odeslání. To docílíme přidáním metody na event *onSuccess* – tento event je spuštěn pouze tehdy, pokud je formulář plně validní a bez chyb. Formulář používá ještě 2 další eventy **onSubmit** a **onError**. Event **onSubmit** se vždy zavolá při submitu (odeslání) formuláře, nezávisle na tom, jestli je nebo není validní. Druhý event, **onError** se zase zavolá, pokud formuláře obsahuje validační chyby. Implementaci logiky formuláře lze provést dvěma způsoby, první je pomocí anonymní metody, což se až tak nedoporučuje a druhá pomocí separátní metody (tento případ je využit ve zdrojovém kódu 1. Doporučuje se spíše druhá možnost, vzniká totiž lepší přehlednost v kódu.[16]

6.3 Databázová vrstva

Pro vývoj větší aplikace, jako je například i tento projekt, je potřeba rozdělit databázovou vrstvu od aplikační. To nám zajistí, obrovskou výhodu při úpravách dotazů. Díky oddělené vrstvě je vždy upravíme na jednom místě a ne na více místech v presenteru, to eliminuje podstatně velkou chybovost, kdy by se na nějaký dotaz například zapomnělo.

Jelikož aplikace stojí na Nette Frameworku, využijeme pro komunikaci s MySQL databází knihovnu Nette Database. Ta se skládá z 2 vrstev. První – Nette Database Core[17] je základní vrstva pro přístup k databázi, tzv. database abstraction layer. Pro připojení k databázi jsme použili registraci pomocí aplikační konfiguraci – viz 2, další obrovská výhoda Nette Frameworku, registrování komponent, modelů a nebo taky třeba i databáze. Nyní už můžeme psát různé SQL dotazy pomocí metody **query()** nad objektem **Nette\Database\Connection**.

```
database:
   dsn: 'mysql:host=localhost;dbname=calculator'
   user: 'root'
   password: 'root'
   options:
      lazy: yes
```

Výpis 2: Připojení databáze pomocí aplikační konfigurace

Druhá, zajímavější, vrstva je Nette Database Explorer.[18] Ten nám zjednodušuje bezpečné, rychlé a efektivní psaní SQL dotazů. Používání Database Explorer začíná od tabulky a to zavoláním metody **table** nad objektem Nette\Database\Context, ta nám vrací důležitý objekt **Selection**. S tím pracují všechny moje metody v použitých repozitářích. Objekt **Selection** má spoustu výhod, lze ním iterovat, bez nutnosti fetchnout (vrácení řádků z výsledku dotazu) – kdy řádky jsou pak instance objektu **ActiveRow**, který lze přirovnat k návrhovému vzoru active record. K **ActiveRow** lze přistupovat jako k poli nebo jako k objektu a současně lze používat příkazy jako například **update()**, **where()** a další, které přistupují k databázi a například upravují případný řádek. Nette Database Explorer používá snad všechny základní metody (insert, update, select, where, order, group), kompletní seznam lze nalézt v dokumentaci. Za zmínku určitě stojí říct, jak funguji databázové JOIN. Takovou metodu totiž Nette Database Explorer neobsahuje, sám si totiž joiny generuje pomocí **ref()** – relace Has One a **related()** – relace Has Many.

V Aplikaci jsem si vytvořil supertřídu **BaseModel**, která zajišťuje připojení k databázi a mapuje název tabulky podle souboru. Díky tomu, mi stačilo vytvořit další repozitáře s názvem souboru shodným jako je název tabulky s camel case syntaxí a moje mapování vše zajistilo. Přidal jsem své další vlastní metody pro lepší fungování jako například **find()**, kde parametr je id tabulky a **findAll()** – to stejné jako *$this->database->table($this->table)*, jen zkráceny tvar.

Další obrovská výhoda používáni Nette Database Exploreru je možnost dosazování parametrů do dotazů a tím zabráněním SQL injection.

6.4 AJAX požadavky

Nette nabízí poměrně snadné zpracování AJAX požadavků a překreslení dat pomocí tzv. **snippetů**. Pro zpracování takových požadavků se používá syntaxe **handle<nazev>()**. Pokud potřebujeme vrátit data a po té je v Javascriptu zpracovat uděláme tak pomocí **sendResponse()**, ve většině případů postačí rovnou **sendJson()** – to vnitřně zavolá **sendResponse()** s **JsonResponse** daty a tím vrátí aplikaci odpověď s daty ve formátu JSON.

Pokud chceme po zpracování nějakého AJAX požadavku, překreslit data, potřebujeme nejdřív oblast, kterou budeme překreslovat obalit do makra **{snippet <nazev>} {/snippet}** nebo zadat do HTML DOMu makro **n:snippet='<nazev>'**. Po té před zpracováním se hodí ověřit, jestli opravdu požadavek přišel AJAXem a to příkazem **isAjax()**. Pak stačí pomocí **redrawControl()** překreslit všechny snippety, pokud chceme překreslit pouze nějaké použijeme syntaxi **redrawControl(<nazev>)**. Zpracování AJAXem a překreslování dat se používá v mojí v celém konfigurátoru i kalkulátoru. Třeba při přecházení do dalších kroků, vždy se odešle AJAX požadavek, ten zpracuje nějaké data, vrátí potřebné data zpět a v Javascriptu se dokreslí potřebné věci. Nebo při změně parametrů v kalkulátoru se vždy překreslí parametry financování. To se řeší pomocí odeslání formuláře AJAXem.

6.5 Api

Jelikož jsou úrokové sazby uloženy v databázi, tak kalkulátor potřebuje mít přístup k nim a pracovat s nimi. Jednoduše by se dalo v aplikační vrstvě volat metody z repozitáře a pracovat s výsledky. Ovšem pro tuto aplikaci jsem využil možnost a vytvořil menší, interní api pro kalkulátor. Api je napsaná v Nette pouze jako presenter, jelikož se jedná o malou, interní api, nebylo potřeba využívat jiné nástroje, programovací jazyky a tím ji oddělovat od projektu. Kalkulátor při změně parametrů zavolá dotaz na api, ještě před tím zkontroluje jestli již tato kombinace parametrů není uložena v cache. Pokud je, tak se databáze, ani api nezatěžuje a vezme se výsledek přímo z cache. Pokud kombinace není uložená, odešle se požadavek na api, ta zpracuje parametry, dotáže se na databázi a tím získá úrokovou sazbu. Dále pomocí interních metod, vypočte jak měsíční splátku vozidla, tak měsíční splátku pojištění. To celé vrátí JSONem do **CalculatorPresenter** odkud dotaz na api přišel. Ten výsledek uloží do cache a dále data zpracuje do šablony, kde už uvidí výsledek uživatel. Pokud by se v administraci změnila úroková sazba, vždy se kontroluje, jestli je uložená v cache, pokud je tak se smaže. Tím zabráníme chybě, kdy by byla uložena stará úroková sazba a nová by se nebrala v potaz.

Pro komunikaci s api jsem si vytvořil komponentu, která má 2 metody, první nejdůležitější **call_api()** – viz 3. Jedná se o univerzální metodu, která má 3 parametry, první je HTTP metoda – POST, GET. Díky tomuto parametru metoda dokáže poznat jestli se jedná o POST a

tím poslat data pomocí JSONu, nebo ne. Druhý je název metody za lomítkem, na kterou se api zavolá. Mini api má pouze jednu metodu – **/Api/installments**. Posledním parametrem jsou data, které chceme poslat v případe POST požadavku. Druhá metoda, viz 4 slouží k zpracování JSON odpovědi z api do pole.

```php
public static function call_api(string $method, string $action, ?array $data =
    null): string
    {
        $url = "https://{$_SERVER["SERVER_NAME"]}/{$action}";
        $curl = curl_init();
        try {
            if (mb_strtolower($method) == "post") {
                curl_setopt($curl, CURLOPT_POST, 1);
                if ($data) {
                    $dataEncoded = json_encode($data);
                    curl_setopt($curl, CURLOPT_POSTFIELDS, $dataEncoded);
                    curl_setopt($curl, CURLOPT_HTTPHEADER, array(
                            'Content-Type: application/json',
                            'Content-Length: ' . strlen($dataEncoded)
                        )
                    );
                }
            } elseif (mb_strtolower($method) == "get" && $data) {
                $url = "{$url}?" . http_build_query($data);
            }

            curl_setopt($curl, CURLOPT_URL, $url);
            curl_setopt($curl, CURLOPT_RETURNTRANSFER, 1);

            $result = curl_exec($curl);
            if ($result === false) {
                $result = '{"status": "error","message": "Api is not currently
                    available"}';
            }
        } catch (\Exception $exception) {
            $result = "{\"status\": \"error\",\"message\": \"{$exception}\"}";
        } finally {
            curl_close($curl);
        }
        return $result;
    }
```

Výpis 3: Univerzální metoda pro volání api požadavků

```
public static function parse_response(string $response): array
    {
        $result = [];
        try {
            $result = Json::decode($response, Json::FORCE_ARRAY);
        } catch (JsonException $e) {

        }
        return $result;
    }
```

Výpis 4: Metoda zpracování JSON odpovědi z api do pole

6.6 Datagridy

V administraci se využívají 2 různé datagridy. První jednoduchý, ryze Javascriptový, se využívá pouze v případě výpisu značek a modelů. Jelikož se jedná o málo dat, nebylo potřeba ze začátku používat jiný. Tento datagrid využívá filtrování výsledků (pomocí fulltext vyhledávání), řazení sloupců a i stránkování. Nevýhodou Javascriptových datagridů je pomalé načítaní velkých dat – již u 1000 záznamů, je načítání viditelné. Aplikace se načítá okolo 2-3 sekund. Čím více dat, tím podstatně pomalejší načítání.

Pro další případy jakožto hlavně správa úrokových sazeb jsem byl nucen použít druhý datagrid – Ublaboo Datagrid.[19] Jedná se o datagrid přímo pro Nette Framework. Také obsahuje filtrování dat – ne fulltext, ale přímo pomocí daného sloupce v databázi. Dále stránkování , řazení a další plno výhod. Největší výhoda je, že si Ublaboo datagrid nebere všechny položky najednou – předává se mu **Selection** od Nette Database. Díky tomu si sám datagrid zpracovává data podle toho, kolik chce uživatel v dané chvíli vidět – ve výchozím nastavení maximálně 50 položek na stránce. Sám si generuje potřebné SQL dotazy s limity a offsety a tím optimalizuje chod aplikace.

7 Testování aplikace

Před spuštěním aplikace na serveru, je potřeba řádně otestovat jestli neobsahuje nějaké chyby. To lze více způsoby, buď napsaní přímo testů, dokonce má Nette pro to i knihovnu, nebo požádat další lidi o otestování aplikace. Pokud aplikaci testuje programátor, který ji vyvinul, dost často se stává, že přehlídne chyby – proto pokud nemá programátor nějakého testera, doporučuje se napsat, vždy před naprogramováním dané problematiky, unit testy, který pak otestuje chod aplikace automatizovaně.

7.1 Způsob testování

Pro testování aplikace jsem oslovil více lidí, většina z nich hlavně testovala konfigurátor s kalkulátorem, aby neměly zbytečný přístup do administrace. Lidi,kteří testovali aplikaci, jsem si rozdělil do různých skupin, aby každá skupina testovala jen jeden typ zařízení – telefon, tablet a notebook nebo stolní počítač. Jsem jim vysvětlil co konfigurátor umí a co by měli získat. Některým jsem udělil i přístup do administrace, abych otestoval přehlednost a také funkčnost.

7.2 Průběh

Do testování se zapojil i pán Ing. Radoslav Fasuga, Ph.D., ten mi jediný dokázal najít v konfigurátoru chybu, zbytek testerů hlásili hlavně chyby s responzivitou. Chyby jsem obratem opravoval a uživatelé mohli testovat dále. Feedback od uživatelů na konfigurátor s kalkulátorem byl hlavně pozitivní jedná se totiž o jednoduché rozhraní, které si hned osvojily. Administrace byla na tom trošku horší, hlavně sekce značka a modely, ale po případném dovysvětlení, jak to má fungovat a kde to mají najít si to také osvojily. To by se dalo vyřešit nějakým jednoduchým návodem kde co v administraci najít a jak ji ovládat. Dále uživatelé navrhovali i případné vylepšení stránky, mezi nejčetnějšími byly fotografie u modelů aut, kdyby byly fotografie tak by chtěli i měnit barvu fotografie podle barvy v konfigurátoru. Další žádosti byly o možném přidání dalších značek s modely.

8 Závěr

Z analýzy jsem zjistil, jak správně funguje financování automobilů a případně jejich pojištění. Pro přesné úrokové sazby, je potřeba znát interní ceník leasingových společností a pravidelně jej aktualizovat kvůli změnám hodnoty peněz, tím by se kalkulátor stál co nejpřesnější. U pojištění to je vždy jinak, každá leasingová společnost má domluvené svoje dealery u různých pojišťoven, kteří jím nabízejí lepší ceny, než by měli normální zákazníci. Další zajímavá věc byla určitě jak funguje segmantace zákazníků čím větší a detailnější segmentace tím přesnější výpočet pojištění. Pro náš kalkulátor stačila základní segmentace, kterou využívají leasingové společnosti v jejich kalkulátorech.

V praktické části jsem naimplementoval vlastní konfigurátor s kalkulátorem i s možností změn v administračním rozhraní. Na základě testování můžu usoudit, že kalkulátor splnil svůj účel, je jednoduchý a hlavně přehledný a to na všech zařízeních. Díky administračního rozhraní lze kdykoliv změnit a upravit úrokové sazby, značky, modely apod. pro konfigurátor.

Pokud bych porovnal svůj konfigurátor se stávajícími řešení například konfigurátor od Škody, tak je určitě jejich více propracovaný, ale povětšinou se jedná už jen o práci s obrázky – změny barev vozidla, změna kol apod. Počítaní financování probíhá dost podobně, ovšem částky se budou lišit, jelikož Škoda používá přesný sazebník jak od leasingové služby tak i od pojišťoven.

Odkazy

1. *Skoda Auto Car Configurator* [online] [cit. 2019-09-29]. Dostupné z: `https://cc.skoda-auto.com/cze/cs-CZ`.
2. *Volkswagen Financial Services kalkulátor* [online] [cit. 2019-09-29]. Dostupné z: `https://form.vwfs.cz/kalkulacka`.
3. *Usetreno.cz: Jak na výpočet úroku z půjčky?* [Online] [cit. 2019-09-29]. Dostupné z: `https://www.usetreno.cz/vypocet-uroku-z-pujcky`.
4. *Splatnost a splácení hypotečního úvěru* [online] [cit. 2019-09-29]. Dostupné z: `https://www.finance.cz/bydleni/hypoteky/abeceda-hypotek/splaceni`.
5. *Klik.cz* [online] [cit. 2019-09-29]. Dostupné z: `https://www.klikpojisteni.cz`.
6. *Povinne-ruceni.com* [online] [cit. 2019-09-29]. Dostupné z: `https://www.povinne-ruceni.com`.
7. *ePojisteni.cz*. Dostupné také z: `https://www.epojisteni.cz`.
8. *Srovnator.cz*. Dostupné také z: `https://www.srovnator.cz`.
9. *Volkswagen Financial Services* [online] [cit. 2019-10-27]. Dostupné z: `https://www.vwfs.cz`.
10. WELLING LUKE, Thomson Laura. *Mistrovství PHP a MySQL*. Brno: Computer Press, 2017. ISBN 978-80-251-4892-1.
11. *Nette – Pohodlný a bezpečný vývoj webových aplikací v PHP* [online] [cit. 2020-02-11]. Dostupné z: `https://nette.org/cs`.
12. *Composer – Wikipedie* [online] [cit. 2019-10-27]. Dostupné z: `https://cs.wikipedia.org/wiki/Composer`.
13. *Bootstrap · The most popular HTML, CSS, and JS library in the world.* [Online] [cit. 2020-04-25]. Dostupné z: `https://getbootstrap.com`.
14. *jQuery* [online] [cit. 2020-04-25]. Dostupné z: `https://jquery.com`.
15. *Presentery | Nette Docs* [online] [cit. 2020-04-20]. Dostupné z: `https://doc.nette.org/cs/3.0/presenters`.
16. *Formuláře | Nette Docs* [online] [cit. 2020-04-20]. Dostupné z: `https://doc.nette.org/cs/3.0/forms#toc-formulare`.
17. *Database Core | Nette Docs* [online] [cit. 2020-04-20]. Dostupné z: `https://doc.nette.org/cs/3.0/forms#toc-formulare`.
18. *Database Explorer | Nette Docs*.

19. *contributte / datagrid - DataGrid for Nette Framework: filtering, sorting, pagination, tree view, table view, translator, etc* [online] [cit. 2020-04-20]. Dostupné z: `https://contributte.org/packages/contributte/datagrid/`.

20. SENDIANG M. Polii A., Mappadang J. Minimization of SQL injection in scheduling application development. 2017, s. 14–20.